Distributed Authorization by Multiparty Trust Negotiation

Charles C. Zhang and Marianne Winslett

University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
{cczhang,winslett}@cs.uiuc.edu

Abstract. Automated trust negotiation (ATN) is a promising approach to establishing trust between two entities without any prior knowledge of each other. However, real-world authorization processes often involve online input from third parties, which ATN does not support. In this paper, we introduce *multiparty trust negotiation* (MTN) as a new approach to distributed authorization. We define a Datalog-based policy language, Distributed Authorization and Release Control Logic (DARCL), to specify both authorization and release control policies. DARCL suits the needs of MTN and can also serve as a powerful general-purpose policy language for authorization. To orchestrate the negotiation process among multiple parties without a centralized moderator, we propose the *diffusion negotiation protocol*, a set of message-passing conventions that allows parties to carry out a negotiation in a distributed fashion. Building on top of the diffusion negotiation protocol, we propose two negotiation strategies, both safe and complete, to drive MTN with different tradeoffs between privacy and negotiation speed.

1 Introduction

Conventional authorization systems are *closed* in the sense that all important properties of a user are known in advance, before the user requests authorization. In open distributed systems like the Web, however, often strangers have to interact with one another to receive or provide services. When two parties interact to reach an authorization decision, automated trust negotiation (ATN) is an effective approach to establishing trust without any prior knowledge of each other. Under ATN, each party has authorization policies based on digital credentials to limit outsiders' access to its sensitive resources. When an outsider requests access to a sensitive resource, the ensuing trust negotiation includes a sequence of bilateral credential disclosures. Less sensitive credentials are disclosed first, to build up enough trust to disclose more sensitive credentials. The negotiation ends when the resource owner's authorization policy for access is satisfied, it becomes clear that trust will not be established, or either party breaks off the negotiation. If trust is established, the resource requester is authorized to access the owner's resource.

Real world authorization decisions often involve more than two parties. For example, when one submits an online application for purchase of automobile insurance, the insurance company needs to evaluate the buyer's driving record and credit scores to reduce the risk of the sale. The authorization process can be decomposed into a number of two-party negotiations, i.e., one negotiation between the insurance company and

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 282–299, 2008.

the buyer for the purchase, another between the insurance company and the Department of Motor Vehicles (DMV) for the buyer's driving record, and a third between the insurance company and the credit bureau for the buyer's credit scores. However, decomposition is insufficient to reduce an automated multiparty authorization into standalone two-party trust negotiations, because individual negotiations may depend on one another, and therefore need to be interleaved in a certain order for the overall negotiation to succeed. For example, the negotiation between the insurance company and the buyer cannot finish until the insurance company gets the result from the negotiation between the insurance company and the credit bureau for the credit report, which in turn requires the insurance company to get the buyer's authorization for a credit check. For such a multiparty authorization, we need a policy language expressive enough to describe the dependencies between the participating parties and their requests, and a systematic and automated way to decompose the authorization into multiple two-party negotiations and interleave them in an order that respects the dependencies. We also need a way to detect cycles of dependencies and handle them appropriately at run time. Overall, we need a new authorization paradigm, *Multiparty Trust Negotiation (MTN)*, where a negotiation can be automatically carried out among multiple parties in accordance with each party's authorization policies.

Copyright, privacy, and security considerations often lead users to restrict the flow of sensitive information. For example, the Health Insurance Portability and Accountability Act (HIPAA) [1] requires hospitals to make a reasonable effort to disclose only the minimum necessary health information to third parties for purposes such as diagnosis and payment. Inappropriate disclosures can cause privacy breaches and serious damage such as identity theft. In such situations, a release control policy is used to specify the conditions under which a piece of information can be disclosed (sent) to another party. Release policies are related to authorization policies, which govern access to arbitrary resources under a "pull" paradigm. For example, an authorization policy decides whether Alice can access a highly classified database, while a release control policy decides who will be told that Alice has such a privilege. Some policy languages clearly distinguish between these two kinds of policies [2,3], and put restrictions on how they can be used together; e.g., one cannot base an authorization decision on a release control condition. In the DARCL policy language proposed in this paper, authorization and release control are naturally integrated: an authorization can be based on a release condition, and vice versa. Further, DARCL allows the policy writer to specify both the source and destination of each disclosure, so that an authorization decision can be based not only on the content of received information, but also on its source. This feature allows DARCL to specify useful policies that other languages cannot, and also makes DARCL a suitable policy language for MTN. For example, DARCL makes it easy to say that before Alice gives Bob access to a certain resource, Bob has to disclose certain of his qualifications to Alice and a third party Carl has to vouch for Bob.

A trust negotiation *protocol* specifies high-level conventions for communication between negotiating parties, including a set of permissible message types. In the most common form of two-party negotiation protocol [4,5,6,7], each party takes turns sending messages until the negotiation succeeds or fails. It is harder to define a protocol that works for more than two parties. If we allow one peer to assume the role as a negotiation

moderator, the moderator can organize the negotiation into multiple *rounds*. In each round, a peer sends requests to others and replies to requests it receives in previous rounds. The negotiation continues until at a certain round, the moderator declares the success or failure of the negotiation. Unfortunately, the centralized control of the moderator approach directly contradicts the autonomous nature of open distributed systems. For example, we need to be able to allow negotiating parties to leave and rejoin the negotiation on the fly, making the moderator's job complex and hard to scale. Ideally, an MTN protocol should be distributed and free of centralized control.

We make the following contributions in this paper:

1. We propose MTN as a new paradigm for establishing trust in open distributed systems, while leveraging the effectiveness of two-party ATN. We extend and redefine concepts and theories developed for ATN, so that we can use them to establish trust among multiple parties.
2. We define DARCL as the first distributed policy language that supports both authorization and release control in a unified way. DARCL policies can take the source and destination of each credential disclosure into consideration, allowing policy-writers to specify finer-grained access control constraints than with existing languages [8,9,10,11].
3. We present a lightweight distributed protocol for MTN, whose decentralized nature makes it well-suited for peer-to-peer environments.
4. We provide two MTN negotiation strategies that are safe and complete, with different tradeoffs between privacy and negotiation speed.

The rest of the paper is organized as follows. In Section 2 we present related work. In Section 3 we define the DARCL policy language and related concepts. In Section 4 we define MTN protocols and strategies, and introduce the diffusion negotiation protocol. We present an eager negotiation strategy in Section 5 and a more cautious strategy in Section 6. We give conclusions and discuss future work in Section 7.

2 Related Work

Since Winsborough et al. first introduced automated trust negotiation [4], a growing body of work has been done in this area. Yu et al. [5,12,13] differentiated negotiation strategies from protocols, proposed ways to limit the disclosure of sensitive credentials, and devised ways to allow each negotiating party to pick its own negotiation strategy autonomously. We build on their work by extending their basic concepts such as disclosure sequences and strategies to work in an environment with more than two parties.

There are prior efforts to apply ATN to establish trust specifically in P2P systems. Ye et al. [14] proposed a collaborative ATN scheme that uses locally trusted third parties (LLTPs) to solve circular policy dependencies during negotiation. One of our goals is to be able to handle certain common types of circular dependencies without relying on trusted third parties. Bertino et al. [15] proposed Trust-X as a comprehensive framework for ATN in a P2P environment. Trust-X offers a number of innovative features, such as trust tickets, to speed up the negotiation. Both of these projects support two-party trust negotiation only.

The research in distributed credential discovery and proof construction is the closest to our work. QCM [16] and its successor SD3 [10] are trust management systems that can automatically and recursively contact a remote party to gather credentials during the policy evaluation. Li et al. [17] designed a goal-directed credential chain discovery algorithm for their RT family of role-based trust-management systems, which was further enhanced to support parameterized roles and constraints [18]. Bauer et al. [19] designed a lazy proof construction algorithm that places the burden of proof on the credential issuers; later they revised this algorithm to improve performance [20] through techniques such as pre-computing delegation chains. Compared to these proof construction algorithms, MTN better supports autonomy for each peer with respect to which message to send, and more naturally fits environments where peers employ customized negotiation strategies based on their own constraints such as privacy sensitivity and resource availability, and only provide best-effort service. A second difference is that MTN bases its authorizations on iterative asynchronous message exchange, instead of the recursive RPC used in distributed proof construction approaches. This simplifies implementation and lowers runtime costs for maintaining the complex state of recursive RPCs. The third difference is that the end result of a proof construction algorithm is a proof of authorization, encoded as a tree or list of rules, facts, and derivation rules, which the resource owner has to verify. Parts of the proof may be repeatedly verified by different peers while the proof is constructed. MTN, however, uses only signed facts in disclosures, and eliminates the need for proof delivery and verification.

Like almost every authorization policy language from the research community, DARCL is based on Datalog [10,8,21,22,7,3]. Similar to the policy languages of PeerTrust [6], Cassandra [11], and PeerAccess [3], DARCL can explicitly model the origin, flow, and distribution of credentials, which is implicit in other languages; yet only DARCL allows explicit specification of constraints on the source and destination of each disclosure, which gives the policy-writer finer-grained control over credential flow and makes DARCL a suitable policy language for MTN.

Multiparty negotiation has been a research topic in multi-agent systems and business intelligence. CONCENSUS [23] uses multiparty negotiation for conflict resolution in concurrent engineering design. Querou et al. [24] presented an iterative method for generating Pareto-optimal solutions in multiparty negotiations. Both of them are based on mathematical models for *quantitative* conflict resolution, while MTN is driven by authorization policies specified in logical formulas.

3 The DARCL Policy Language

We model a distributed system as a finite set of peers, each possessing a finite knowledge base, who communicate with each other by exchanging messages about their resources and services. Following the popular trend in trust management [10,8,21,22], we propose a declarative policy language, DARCL, based on Datalog. The syntax of DARCL is given in Figure 1. A peer's authorization policy for access to a resource or service consists of a set of DARCL rules specifying the conditions that must be satisfied before the access to the service can be granted.

$rule$ ::= $disclosure \leftarrow disclosure \ \wedge \ \cdots \wedge \ disclosure$
$disclosure$ ::= $peer \uparrow credential \downarrow peer$
$credential$ ::= $peer.credential\_name(term, \ \ldots, \ term)$
$term$ ::= $peer \mid value$
$peer$ ::= $variable \mid peer\_name$
$value$ ::= $variable \mid$ string
$variable$ ::= $x \mid y \mid \ldots$
$peer\_name$ ::= $Alice \mid Bob \mid \ldots$
$credential\_name$ ::= string

Fig. 1. Syntax of DARCL

In DARCL, credentials are signed predicates, which can be used to represent attributes of subjects and authorizations for access to resources and services. DARCL abstracts away several properties of real-world credentials such as X.509 certificates, retaining only those needed to reason about distributed authorization. In the credential $University.isRegisteredStudent(Alice)$, $University$ is the *issuer* who signs the credential, Alice is the *subject*, and $isRegisteredStudent$ is the credential name. A credential can have more than one subject. In this paper, all DARCL credential names and peer names will have more than one letter, to distinguish them from variables such as x and y; and peer names will not be used as string-valued terms. We can *instantiate* a variable in a DARCL formula by replacing all its occurrences with a constant. If a DARCL formula does not contain any variables, then it is *ground*.

We use $peer_0 \uparrow \mathcal{C} \downarrow peer_1$ to denote a *disclosure*, the sending of a message from one peer to another, where $peer_0$ is the *source* and $peer_1$ is the *destination* of the message, and $\mathcal{C}$ is its content. The head of each DARCL rule contains a disclosure. A rule's body can be empty, in which case the head of the rule is a *fact*. Intuitively, $Alice \uparrow \mathcal{C}_0 \downarrow Bob$ in a rule's head means Alice authorizes the disclosure of $\mathcal{C}_0$ to Bob, and $Bob \uparrow \mathcal{C}_1 \downarrow Alice$ in a rule's body means Alice has received $\mathcal{C}_1$ from Bob. DARCL rules must also satisfy three additional constraints, which simplify rule specification and prevent certain nonsensical policies. Let $S_0 \uparrow \mathcal{C}_0 \downarrow D_0 \leftarrow S_1 \uparrow \mathcal{C}_1 \downarrow D_1, \ \ldots, \ S_n \uparrow \mathcal{C}_n \downarrow D_n$ be a rule in Alice's authorization policy.

1. For every $i \in [0, n]$, either S_i or D_i must be Alice; when the rule's body is not empty, the source S_0 of the disclosure in the rule's head must be Alice. This is because Alice can only base her authorization decisions on her local state, i.e., her own policies plus the credentials that have been sent to her. What kind of credential another party Bob sends to Carl is beyond Alice's knowledge; similarly, Alice cannot force Bob to disclose a credential.
2. If the issuer of the credential $\mathcal{C}_0$ in the head is not Alice and the rule's body is not empty, then $\mathcal{C}_0$ must be one of the credentials $\mathcal{C}_i$ in the body, $1 \leq i \leq n$. In other words, before Alice can disclose a credential signed by somebody else, she must have received it.
3. Every variable in the rule body must also occur in the head. Under this constraint, if the disclosure in the head of a rule is ground after variable instantiations, so are the disclosures in the body of the rule. As enforced in our inference rules shown later, at run time every disclosed credential will be ground.

When the source and destination of a disclosure are the same peer, e.g., $Alice \uparrow \mathcal{C} \downarrow Alice$, we call it a *singular-disclosure*. We use the singular-disclosure $Alice \uparrow \mathcal{C} \downarrow Alice$ to represent the fact that Alice possesses the credential $\mathcal{C}$. If the issuer of $\mathcal{C}$ is not Alice, $Alice \uparrow \mathcal{C} \downarrow Alice$ means Alice has received $\mathcal{C}$. If Alice is the issuer of $\mathcal{C}$, then $Alice \uparrow \mathcal{C} \downarrow Alice$ means Alice is willing to sign or has signed $\mathcal{C}$, depending on whether it is in the head or body of the rule. For simplification, we abbreviate a singular-disclosure $Alice \uparrow \mathcal{C} \downarrow Alice$ as just $\mathcal{C}$. This simplification causes no confusion, because restriction 1 tells us that $\mathcal{C}$ in Alice's knowledge base must stand for $Alice \uparrow \mathcal{C} \downarrow Alice$, and not $Bob \uparrow \mathcal{C} \downarrow Bob$. If a disclosure's source and destination are different, we call it a *remote* disclosure. If an authorization rule's head is a remote disclosure, then it is a *release control rule*. The collection of all a peer's policies and credentials is its *knowledge base*.

In practice, credentials are often stored with their issuers or their subjects [17,25]; in such a case, the source of a disclosure can be constrained to be the credential's issuer or one of its subjects. Such constraints are easy to express in DARCL but not built into the language, because we want to support additional scenarios, such as when the distribution of credentials is outsourced to a third party.

3.1 Policy Examples

We use two examples to show how DARCL can be used to specify release control and authorization policies.

Example 1 Consider the following set of rules in Alice's knowledge base:

(1) $Bob.trusts(Carrie) \leftarrow Bob \uparrow Bob.trusts(Carrie) \downarrow Alice$
(2) $Alice \uparrow Bob.trusts(Carrie) \downarrow Diana \leftarrow Bob.trusts(Carrie)$
(3) $Alice \uparrow Bob.trusts(Carrie) \downarrow x \leftarrow Bob \uparrow Bob.trusts(Carrie) \downarrow Alice \wedge$
$Carrie \uparrow Bob.trusts(Carrie) \downarrow Alice$
(4) $Alice.trusts(x) \leftarrow Diana.trusts(x)$
(5) $Alice \uparrow Alice.trusts(Diana) \downarrow x \leftarrow Alice.trusts(x)$

Rule (1) is trivial: it says that if Bob tells Alice that he trusts Carrie, Alice knows that he trusts Carrie. Rule (2) says that if Alice knows that Bob trusts Carrie, Alice will tell Diana about it. Rule (3) says that if both Bob and Carrie tell Alice that Bob trusts Carrie, Alice will tell everybody about it; Alice's intuition is that since both Bob and Carrie are open with her about it, they probably do not treat it as a secret. Rule (4) says that Alice will trust anyone that she knows that Diana trusts. Rule (5) says that Alice will tell everyone she trusts about the fact that she trusts Diana. The difference between a singular-disclosure $d = Alice.trusts(Diana)$ and a regular disclosure $Alice \uparrow d \downarrow Eddie$ is clear: the former states that Alice trusts Diana, while the latter states to whom Alice discloses that fact. Note that if we restrict the rules to use singular-disclosures only, DARCL shrinks to a variant of existing Datalog-based authorization languages, such as those used in [8,9,10]. These languages deal with authorization but not release control, thus cannot specify rules like (3).

Example 2 Suppose Alice, a Canadian national, would like to apply for a digital Mexican visa so that she can enter Mexico as a tourist. For security considerations, the Mexican government has a new policy that requires its visa department, Embassy of Mexico (EM), not to issue a visa for any Canadian national unless the applicant presents a valid Canadian passport and passes the background check of Mexico's security agency, Direccion Federal de Seguridad (DFS). To respect privacy, DFS will not release Alice's background check result to EM unless DFS gets Alice's permission to do so first. For convenience and security considerations, DFS, as a security agency, deals directly with EM, but not directly with any individual; therefore, any communication between Alice and the DFS has to go through EM. Alice is willing to give EM what it requests, provided that she can verify that EM has a digital certificate signed by the Mexican government (MG) to show that EM really is the official Mexican embassy. EM, on the other hand, is willing to show its certificate signed by MG to anyone. In DARCL, these policies are as follows:

EM:

$$EM \uparrow EM.visa(x) \downarrow x \leftarrow x \uparrow Canada.passport(x) \downarrow EM \wedge x \uparrow x.OkToRelease(DFS, EM) \downarrow EM \wedge DFS \uparrow DFS.clear(x) \downarrow EM$$

$$EM \uparrow x.OkToRelease(DFS, EM) \downarrow DFS \leftarrow x \uparrow x.OkToRelease(DFS, EM) \downarrow EM$$

$$EM \uparrow MG.officialEmbassy(EM) \downarrow x$$

DFS:

$$DFS \uparrow DFS.clear(x) \downarrow EM \leftarrow EM \uparrow x.okToRelease(DFS, EM) \downarrow DFS \wedge DFS.clear(x)$$

Alice:

$$Alice \uparrow Alice.okToRelease(DFS, x) \downarrow x \leftarrow x \uparrow MG.officialEmbassy(x) \downarrow Alice$$

$$Alice \uparrow Canada.passport(Alice) \downarrow x \leftarrow x \uparrow MG.officialEmbassy(x) \downarrow Alice \wedge Canada.passport(Alice)$$

We will revisit these examples in subsequent sections.

3.2 Inference Rules

Our next step is to show how authorization decisions can be made based on DARCL policies. Sometimes a peer can make an authorization decision based on only its local authorization policies. Other times, an authorization decision is collectively based on the peer's local policies and the information it receives from other peers. Accordingly, DARCL has two types of inference rules, *local* and *global*. From a logical perspective, the local inference rules are used to derive new facts or rules within a peer's knowledge base, while the global inference rules, together with the local inference rules, can be used to derive new facts based on more than one peer's knowledge base.

Local Inference Rules A peer A can use the following local derivation rules.

- **Instantiation.** From a rule r in A's knowledge base, derive an instance of r by replacing all occurrences of the same variable in r with another variable or literal.
- **Knowledge.** From $B \uparrow d \downarrow A$, derive $A \uparrow d \downarrow A$.
- **Modus ponens.** From a rule $d_0 \leftarrow d_1 \wedge \cdots \wedge d_n$ and facts d_i, $1 \leq i \leq n$ in A's knowledge base, derive d_0.

Global Inference Rule

– From $A \uparrow d \downarrow B$ in A's knowledge base, where d is ground, derive $A \uparrow d \downarrow B$ in B's knowledge base.

Returning to Example 1, let us see how we can use the inference rules to decide whether Alice can tell Edward that she trusts Diana. Suppose Diana trusts Edward and has told Alice about it; i.e., Alice's knowledge base has $Diana \uparrow Diana.trusts(Edward) \downarrow Alice$. By the knowledge rule, we derive $Diana.trusts(Edward)$. Applying instantiation and modus ponens on rule (4), we get $Alice.trusts(Edward)$. Repeating the same derivations on rule (5), we get $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$; i.e., Alice can tell Edward that she trusts Diana.

If we apply both the local and global inference rules to all policies in everyone's knowledge bases, we can reason about every possible authorization that any peer can ever make. Continuing with Example 1, if we apply the global inference rule on the fact $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$ in Alice's knowledge base, we derive $Alice \uparrow Alice.trusts(Diana) \downarrow Edward$ in Edward's knowledge base.

Although it is theoretically interesting to reason about authorizations globally, in practice there is no omniscient authority to reason about everyone's policies in an open distributed system. Before Alice approves an authorization request from Bob, she may request credentials from Bob and other peers, which can trigger more rounds of credential requests and disclosures. MTN is such a process, where peers conduct multilateral message exchanges to collectively make an authorization decision without losing each peer's autonomy; i.e., at any moment, each peer's authorization decisions are still based on its current local policies.

DARCL also has a declarative semantics, which interested readers can find in an extended version of this paper [26].

3.3 Disclosure Sequences

We say a ground disclosure $d = Alice \uparrow \mathcal{C} \downarrow Bob$ is *unlocked* if d is already in Alice's knowledge base, or d can be derived in Alice's knowledge base using the local inference rules; otherwise, d is *locked*. Suppose d is locked, and will be unlocked if Alice receives disclosures $d_1, \ldots, d_n$ from other peers; we say d is unlocked by $d_1, \ldots, d_n$. We say a ground disclosure e is a *relevant* disclosure for d if e derives e by the knowledge rule, or there exists an instantiation r' of rule r that has d as its head and e is a disclosure in the body of r'. If e is a relevant disclosure for d, then every relevant disclosure for e is also a relevant disclosure for d.

A ground disclosure is *safe* if it is unlocked at the time it takes place. For example, Alice can safely disclose $\mathcal{C}$ to Bob if $Alice \uparrow \mathcal{C} \downarrow Bob$ is unlocked in Alice's knowledge base. We define a *disclosure sequence* Seq as $[d_1, \ldots, d_n]$, where each $d_i = S_i \uparrow \mathcal{C}_i \downarrow D_i$ is a remote disclosure representing S_i disclosing $\mathcal{C}_i$ to D_i, and each d_{i+1} takes place after d_i. Seq is a *safe disclosure sequence* for d_n (or simply *safe*) if each d_i is safe at the time it takes place. More specifically, every d_i must either be unlocked before the first disclosure in S takes place, or be unlocked once d_{i-1} has taken place. When Seq is safe, $[d_1, \ldots, d_i]$ is a safe disclosure sequence for d_i, which gives us the following proposition.

```
PARTICIPATEINMTN () {
  while willing to participate do
    if the incoming message queue is empty then
      Wait a short period of time for new messages
    if there is a new message in the incoming message queue then
      Choose such a message m and remove it from the queue
      Record m's receipt time as the current time
      ProcessMessage(m)
}

/* message handler */
PROCESSMESSAGE (m) {
  /* M_received and M_sent store received and sent messages respectively */
  M_received = M_received ∪ {m}
  If m is a disclosure, add m to the local knowledge base L
  Apply the local negotiation strategy with parameters M_received, M_sent, and L,
  which returns a list of messages M
  /* send the messages to their intended recipients */
  if M is not empty then
    for every message k in M do
      Send k to its specified recipient
      Record k's sending time as the current time
      M_sent = M_sent ∪ {k}
}
```

Fig. 2. The Diffusion Negotiation Protocol

Proposition 1. *If* $[d_1, \ldots, d_n]$ *is a safe disclosure sequence, then there is a safe disclosure sequence for* d_i *with at most* i *disclosures, for every* $1 \leq i \leq n$.

The existence of a safe disclosure sequence S for $d = Bob \uparrow \mathcal{C} \downarrow Alice$ means that if the disclosures take place in the order given in S, resource owner Bob can eventually safely grant Alice access to resource $\mathcal{C}$, without violating any peer's authorization policy. Suppose that in Example 2, DFS clears Alice's background by signing $DFS.clear(Alice)$; then there is the following safe disclosure sequence that leads to EM sending a signed visa to Alice.

$[\, EM \uparrow MG.officialEmbassy(EM) \downarrow Alice,\ Alice \uparrow Canada.passport(Alice) \downarrow EM,$
$Alice \uparrow Alice.okToRelease(DFS, EM) \downarrow EM,\ EM \uparrow Alice.okToRelease(DFS, EM) \downarrow DFS,$
$DFS \uparrow DFS.clear(Alice) \downarrow EM,\ EM \uparrow EM.visa(Alice) \downarrow Alice\,]$

The goal of MTN is to find such a safe disclosure sequence.

4 MTN Protocols and Strategies

In our authorization framework, peers negotiate with each other by sending messages, following the conventions specified by a negotiation protocol. Protocols can be specified at different levels. At the lowest level, a protocol defines how messages can be

encoded and transferred through a particular medium. For our purpose of trust negotiation research, we are primarily concerned with high-level message-passing conventions regarding how to start a negotiation, when it is a particular peer's turn to send a message, what the formats of the messages are, and how to tell whether a negotiation succeeds or fails. On top of a common negotiation protocol, a peer employs a negotiation *strategy*, which is its plan of action to achieve a certain goal, e.g., reaching a successful conclusion to the negotiation as soon as possible. More specifically, a negotiation strategy decides the content of each message, i.e., what messages to send back in reply to a received message.

We propose a completely distributed MTN protocol that allows the negotiation to proceed without any centralized control. Party A starts a negotiation by sending a ground request $r = ?B \uparrow R \downarrow A$ to another party B. We call A the *originator* and r the *originating request* of the MTN. After the originating request is sent, no peer sends any message as part of this negotiation, unless it first receives a message. Once a party receives a message, it sends a finite number (possibly zero) of messages to other parties, and then remains "silent" until it receives another message. This protocol for MTN falls into the general class of protocols that Dijkstra and Scholten describe as *diffusing computation* [27], provided that the number of messages that each party sends within a single negotiation is finite, which always has to be true for the negotiation to be useful. We therefore call this MTN protocol the *diffusion protocol*. An MTN *succeeds* when the requested disclosure in the originating request is actually made (e.g., when Bob actually grants Alice access to his resource). An MTN *terminates* when none of the parties sends or receives any more messages.

We give the pseudocode for the diffusion protocol in Figure 2. A peer willing to participate in an MTN is either waiting for messages from other peers, or processing received messages. Incoming messages are put in a queue until processed. The choice of which queued message to process next is a strategic decision; for the purpose of this paper, any choice is satisfactory (FIFO, LIFO, random, giving higher priority to certain peers or message types, etc.).

Each incoming message is stamped with a receipt time when it is processed, and each outgoing message is stamped with a sending time. Both timestamps represent the party's local time; in other words, we do not assume a globally consistent time clock. We do assume that if a peer Alice sends (respectively, processes) message 1 and later sends (resp. processes) message 2, then Alice's timestamp for message 1 is earlier than her timestamp for message 2. Messages already processed or being processed are stored in the set $M_{received}$, while those already sent to others are stored in the set M_{sent}. A message can be a disclosure d, a request for d (denoted $?d$), a denial to disclose d (denoted as $!d$), or any other type of message that is specific to the strategy that the participating parties adopt. To process a new message m, the local party P_{this} adds m to $M_{received}$, and calls its local negotiation strategy with $M_{received}$, M_{sent}, and its local authorization rules L. The negotiation strategy returns a list of messages, which P subsequently sends to the appropriate recipients. The diffusion protocol is strategy-neutral, meaning that different MTNs can use different strategies while following the same protocol.

We make the following assumptions and simplifications. For clarity, the strategy in Figure 2 is written as though only one negotiation takes place at a time. To support multiple concurrent negotiations, the originating request should include a new globally unique session ID, and the protocol should associate that session ID with each message sent as part of the negotiation. Similarly, the message handler PROCESSMESSAGE and the negotiation strategies given in subsequent sections should all be parameterized with the session ID so that they deal each negotiation separately. While we do not explicitly deal with lost or delayed messages, in practice they can be detected through predefined timeouts and handled accordingly. We require that peers communicate through secured channels, and all the credentials exchanged are digitally signed, thus verifiable, nonforgeable, and nonrepudiable.

5 Eager Strategies

A party can adopt an eager strategy if it is eager to bring the negotiation to a successful conclusion as soon as possible. To speed up the negotiation, an eager strategy aggressively requests relevant remote disclosures and is willing to make requested disclosures as soon as they become unlocked. The first eager strategy that we present is a relatively unsophisticated version, which we call the *basic eager strategy (BES)*.

5.1 Basic Eager Strategy

Figure 3 gives the BES strategy. For a participating peer P_{this}, the goal is to calculate D_{new}, the set of unlocked disclosures that are requested by other parties, but not disclosed yet; and Q_{new}, the set of disclosures that P_{this} would like to request from other parties. The current message m has the latest timestamp in $M_{received}$. If m is a disclosure d, we calculate $D_{unlocked}$, the set of disclosures that are unlocked by d and other previously received disclosures. Since there is no need to disclose unrequested credentials or make the same disclosure to the same peer twice, we intersect $D_{unlocked}$ and the disclosures $Q_{received}$ that other parties requested from P_{this}, then subtract D_{sent}, which contains all disclosures already sent, and finally get D_{new}. If m happens to be a request for disclosure d that is unlocked already, we can simply set D_{new} to be $\{d\}$. If, however, d is still locked, we calculate the set $D_{relevant}$ of all relevant remote disclosures for d, then subtract all disclosures P_{this} received and all disclosures P_{this} requested from others, which gives us Q_{new}, the new disclosures that P_{this} will request from others in order to unlock d. Adding the disclosures in D_{new} and the requests for the disclosures in Q_{new}, we get the messages that P_{this} will send as a response to m.

We are particularly interested in two basic properties of a strategy: *safety* and *completeness*. A strategy is *safe* if every disclosure in the negotiation is safe. Assume that every peer involved in the negotiation follows the same strategy Θ and is willing to participate, and there is no loss of messages. Then strategy Θ is *complete* if the negotiation succeeds whenever there is a safe disclosure sequence for the originating request.

Theorem 1. *The BES strategy is both safe and complete.*

Due to page limitations, we omit all proofs from this paper; interested readers can find them in an extended version of this paper [26].

```
1.  BASICEAGERSTRATEGY (M_received, M_sent, L) {
2.     Let P_this be the current peer
3.     m = the latest message in M_received
4.     Q_sent = set of disclosures P_this requested from others
5.     Q_received = set of disclosures others requested from P_this
6.     Q_new = ∅
7.     D_sent = set of disclosures P_this sent to others
8.     D_received = set of disclosures P_this received from others
9.     D_new = ∅
10.
11.    if m is a disclosure d then
12.       /* Calculate new disclosures D_new that P_this will send to other parties */
13.       D_unlocked = all disclosures unlocked by d and other disclosures in D_received
14.       D_new = D_unlocked ∩ Q_received − D_sent
15.    else if m is a request for disclosure d then
16.       if d is already unlocked then
17.          D_new={d}
18.       else
19.          /* Calculate new disclosures Q_new that P_this will request from others */
20.          D_relevant = all relevant remote disclosures for d
21.          Q_new = D_relevant − D_received − Q_sent
22.
23.    Return the list of messages composed of disclosures in D_new and requests for
       disclosures in Q_new
24. }
```

Fig. 3. The Basic Eager Strategy

5.2 Full Eager Strategy

BES guarantees every negotiation to succeed if the negotiation has a safe disclosure sequence. If, however, there is no safe disclosure sequence, the original requester will not hear anything back about its originating request, which also means that it is unable to decide when to declare that the negotiation has failed. Presumably, we could solve this issue by requiring the participating parties to respond with an explicit denial message when the requested disclosure cannot be made. For example, if Alice finds no rule that can be used to unlock the originating request she receives, she can just explicitly deny this request. The decision to deny a request, nonetheless, is not always easy to make. Suppose Alice's decision on whether to disclose d_1 depends on whether Bob discloses d_2, Bob's decision on d_2 depends on whether Carl discloses d_3, and Carl's decision on d_3 depends on whether Alice discloses d_1. With such a circular dependency, if nobody makes a decision until he or she hears from the dependent party, the negotiation gets *deadlocked*. When a deadlock happens and no one sends or receives any messages, the subnegotiation that spawned the deadlock cycle has effectively terminated. In the absence of deadlock, an MTN under the BES strategy also always terminates. This is because the set of peers is finite and BES does not repeat messages: each request or

```
1. FULLEAGERSTRATEGY (M_received, M_sent, L) {
2.    Let P_this be the current peer
3.    m = the latest message in M_received
4.    M_1 = ∅
5.    /* A data message is a disclosure or a disclosure request */
6.    if m is a data message then
7.       M_1 = BasicEagerStrategy(M_received, M_sent, L)
8.    else
9.       m must be an ACK message &e, for some e ∈ M_sent. Mark e as ACKed.
10.
11.   M_2 = ∅
12.   T = the set of all data messages that P_this received and has not ACKed yet.
13.   if all data messages that P_this sent have been ACKed or P_this is the originator
      then
14.      add to M_2 an ACK message for every message in T
15.   else
16.      add to M_2 an ACK message for every message in T, except the one with the
         earliest receipt time
17.
18.   Return M_1 ∪ M_2
19. }
```

Fig. 4. The Full Eager Strategy

disclosure is sent from one party to another at most once and there are only finitely many potential relevant queries and disclosures. Given the completeness of BES, the original requester can declare failure of the negotiation if the originating request has not been granted when the negotiation terminates. So the problem of detecting the failure of an MTN under BES can be reduced to the detection of the termination of the MTN.

Dijkstra and Scholten give a signaling scheme [27] that can detect the termination of a diffusion computation. Their signaling scheme tracks the balance of messages and signals on each edge between two nodes and centers around a number of invariants on these balances. By enhancing and simplifying their signaling scheme to match the characteristics of MTN, we provide a simple acknowledgment (ACK) scheme that can be superimposed on top of BES and detect the termination of an MTN. This results in an extension of BES, which we call the Full Eager Strategy (FES). We use two types of messages in FES, data messages and ACK messages. A data message is either a disclosure or a disclosure request, as used in BES. An ACK message ($\&m$) acknowledges (abbreviated as ACKs) a data message m. Each data message gets ACKed exactly once in FES. A peer's state is *disengaged* if all the data messages it sent have been ACKed and it has ACKed all the data messages it received; otherwise, its state is *engaged*.

Figure 4 gives the FES strategy. For a newly received message m, we first check its type. If m is a data message, we apply the BES strategy, and get a set of messages M_1 that the current peer P_{this} will send out. On the other hand, if m is an ACK message for a data message e that P_{this} sent out earlier, we mark e as ACKed. We then calculate

the set of data messages that P_{this} is going to ACK. If every data message that P_{this} sent out has been ACKed or P_{this} is the originator of the MTN, we ACK all messages in T, the set of data messages received by P_{this} and not ACKed yet. If not all data messages P_{this} sent have been ACKed, P_{this} ACKs all in T, except the one that has the earliest receipt timestamp.

Theorem 2. *The FES strategy is both safe and complete.*

Theorem 3. *In every MTN under FES, the originator's state will eventually become disengaged. At that point, the negotiation has terminated.*

6 Cautious Strategy

As the eager strategies aggressively explore possible routes to speed up the negotiation by requesting all relevant remote disclosures at the same time, some pending requests become unnecessary and irrelevant when their alternatives are successfully explored. Consequently, the participating parties may send more messages to one another than strictly necessary. For example, when all parties are willing to participate, FES will eventually find all proofs that the originating request holds, rather than stopping and canceling all pending requests once it finds the first successful proof. Since credentials can contain sensitive and valuable information, some parties will place a high priority on their privacy and would prefer to disclose fewer of their credentials, even at the cost of increased negotiation time.

To meet these needs, we propose the cautious strategy, which aims to reduce credential disclosures by making fewer requests in the first place. Under the cautious strategy, when Alice receives a request for a disclosure d that is still locked, she selects only one relevant remote disclosure to request from another party, instead of concurrently requesting all relevant remote disclosures that are still missing in her knowledge base. If the selected disclosure request gets denied, she requests another relevant remote disclosure, and repeats the process until she runs out of options; at that point she explicitly denies the request for d. Since the MTN is distributed among multiple parties, Alice may eventually have sent multiple unanswered requests, and special care must be taken to avoid circular dependencies and prevent deadlock.

Figure 5 describes the cautious strategy. We first examine the type of the message m newly received by the current peer P_{this}. If m is a request for disclosure d, we record this information in e and save e for later reference. If m is a disclosure d or a denial for disclosure $!d$, we examine P_{this}'s received messages to find the disclosure e that P_{this} was trying to unlock at the time that it requested d. If no such e exists, d must be the disclosure in the originating request; in this case, since the originating request has been answered, we can tell whether the negotiation has succeeded or failed. In other cases, we need to continue to process the request for disclosure e. If e is unlocked already, we add it to the return message so that it gets subsequently disclosed. If e is still locked, we examine e's relevant remote disclosures to find those that P_{this} can potentially request. Let q be the latest request for e in P_{this}'s received messages, and f be any of e's relevant remote disclosures that are not present in P_{this}'s knowledge base.

```
1.  CautiousStrategy (M_received, M_sent, L) {
2.     Let P_this be the current peer
3.     m = the latest message in M_received
4.
5.     if m is a request for a disclosure d then
6.        e = d
7.     else
8.        m must be a disclosure d or a denial !d.
9.        Let ?e be the latest request in M_received that has not been denied or disclosed,
          and for which d is relevant.
10.       if no such ?e exists then
11.          /* The originating request of the negotiation must be for d. If m is a dis-
             closure, the MTN has succeeded; otherwise, m is a denial message, and
             the MTN has failed. */
12.          Return ∅
13.    if  e is already unlocked then
14.       Return {e}
15.    Let S be the set containing all remote disclosures f such that (1) f is relevant to
       e, (2) f has not been received by P_this, (3) if P_this has requested f, that request
       has been denied; and (4) P_this has not requested f since it received ?e
16.    if S is empty then
17.       /* There are no more disclosures that P_this can request to unlock e */
18.       Return { !e }
19.    Pick one disclosure g from S
20.    Return {?g}
21. }
```

Fig. 5. The Cautious Strategy

If we requested f already and have not received a response to that request, we should not request f again, as otherwise a cyclic dependency is established. If f was requested after q's receipt time and denied already, there is no need to repeat the request for f, because the request for f will be denied again. If there are no more relevant remote disclosures to request, we have to deny the request for e.

Line 19 of the cautious strategy involves a strategic decision. Based on Alice's past experience, if she thinks that a received request e is likely to be denied eventually, she can choose to request the relevant remote disclosures for e that are most likely to be denied, to minimize the expected amount of effort that she must expend before she can deny e. If she expects that e will not be denied and she wants to grow her knowledge base, she might prefer to send as many requests as possible (to gather as much new information as possible) before concluding that e holds. Under this approach, she needs to delay making new requests that are likely to unlock e, until she has made as many other relevant requests as possible.

Theorem 4. *The cautious strategy is both safe and complete. Further, every negotiation under the cautious strategy eventually terminates, with the original request either denied or disclosed (granted).*

7 Conclusions and Future Work

To allow trust to be established between more than two parties, we have proposed MTN as a new approach to distributed authorization. MTN extends and reinvents the core concepts and theories of ATN, while also addressing the new challenges of coordinating interleaved communication between parties, detecting circular dependencies, and providing scalability in a fully decentralized environment. Our solution approach addresses all key aspects of MTN, including the policy language, negotiation protocols, and strategies. The DARCL policy language is designed for MTN, yet can also serve as a general purpose policy language. DARCL policies can base authorization decisions on credential distributions, allowing the policy writer to specify finer-grained security constraints than in other policy languages, without loss of flexibility. Our diffusion negotiation protocol provides a lightweight, effective set of communication conventions that supports a fully distributed approach to MTN, without relying on trusted third parties to coordinate the negotiation. Our eager and cautious negotiation strategies are safe and complete, with different tradeoffs between privacy and speed: the eager strategy is willing to disclose more credentials than the cautious strategy, in order to speed up the negotiation.

Our solution approach for MTN can be enhanced in a number of aspects. First, strangers currently need a pre-negotiation stage to reach an agreement on what MTN negotiation strategy to use, because different MTN negotiation strategies do not interoperate with each other. We are interested in finding negotiation strategies that respect the autonomy of each party while also guaranteeing completeness. Second, if a party sends a denial message under the cautious strategy, the recipient can guess that the denying party may not have the requested credential. Li and Winsborough [28,29] investigated credential information leakage problems and proposed acknowledgement policies as a way to prevent unauthorized requesters from guessing sensitive information. One could extend the acknowledgment policy approach to work with MTN. Third, it would be interesting to do a performance study regarding the communication complexity, timing properties, as well as the impact of message loss and dynamic arrival and departures of the parties. Finally, for particularly sensitive credentials, additional protection could be provided by extended versions of cryptographic techniques such as hidden policies and credentials [30,31], which greatly reduce the risk of interacting with strangers.

Acknowledgements

This research was supported by NSF under grants IIS-0331707 and CNS-0325951. The authors thank Nikita Borisov and the anonymous reviewers for their valuable comments and suggestions.

References

1. URL: Health insurance portability and accountability act. Web Site (August 1996), `http://www.hhs.gov/ocr/hipaa/`
2. Bonatti, P., Samarati, P.: Regulating Service Access and Information Release on the Web. In: 7th ACM Conference on Computer and Communications Security, Athens (November 2000)

3. Winslett, M., Zhang, C.C., Bonatti, P.A.: PeerAccess: a logic for distributed authorization. In: 12th ACM Conference on Computer and Communications Security, Alexandria, VA, pp. 168–179 (2005)
4. Winsborough, W.H., Seamons, K.E., Jones, V.E.: Automated trust negotiation. In: DARPA Information Survivability Conference and Exposition (January 2000)
5. Winslett, M., Yu, T., Seamons, K.E., Hess, A., Jacobson, J., Jarvis, R., Smith, B., Yu, L.: The TrustBuilder architecture for trust negotiation. IEEE Internet Computing 6(6) (2002)
6. Gavriloaie, R., Nejdl, W., Olmedilla, D., Seamons, K., Winslett, M.: No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web. In: European Semantic Web Symposium (2004)
7. Koshutanski, H., Massacci, F.: An interactive trust management and negotiation scheme. In: Formal Aspects in Security and Trust, pp. 115–128 (2004)
8. DeTreville, J.: Binder, a logic-based security language. In: IEEE Symposium on Security and Privacy, Oakland, CA (2002)
9. Li, N., Mitchell, J.: RT: A role-based trust-management framework. In: Third DARPA Information Survivability Conference and Exposition (April 2003)
10. Jim, T.: SD3: A trust management system with certified evaluation. In: IEEE Symposium on Security and Privacy (2001)
11. Becker, M.Y., Sewell, P.: Cassandra: distributed access control policies with tunable expressiveness. In: 5th IEEE International Workshop on Policies for Distributed Systems and Networks, Yorktown Heights (June 2004)
12. Seamons, K., Winslett, M., Yu, T., Yu, L., Jarvis, R.: Protecting privacy during on-line trust negotiation. In: 2nd Workshop on Privacy Enhancing Technologies, San Francisco, CA (April 2002)
13. Yu, T., Winslett, M., Seamons, K.E.: Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. ACM Transactions on Information and System Security 6(1) (2003)
14. Ye, S., Makedon, F., Ford, J.: Collaborative automated trust negotiation in peer-to-peer systems. In: 4th International Conference on Peer-to-Peer Computing, Washington, DC, USA, pp. 108–115. IEEE Computer Society, Los Alamitos (2004)
15. Bertino, E., Ferrari, E., Squicciarini, A.C.: Trust-X: A peer-to-peer framework for trust establishment. IEEE Transactions on Knowledge and Data Engineering 16(7), 827–842 (2004)
16. Gunter, C.A., Jim, T.: Policy-directed certificate retrieval. Software Practice and Experience 30(15), 1609–1640 (2000)
17. Li, N., Winsborough, W., Mitchell, J.: Distributed credential chain discovery in trust management. Journal of Computer Security 11(1) (February 2003)
18. Mao, Z., Li, N., Winsborough, W.H.: Distributed credential chain discovery in trust management with parameterized roles and constraints (short paper). In: International Conference on Information and Communications Security, pp. 159–173 (2006)
19. Bauer, L., Garriss, S., Reiter, M.K.: Distributed proving in access-control systems. In: IEEE Symposium on Security and Privacy, Berkeley (May 2005)
20. Bauer, L., Garriss, S., Reiter, M.K.: Efficient proving for practical distributed access-control systems. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 19–37. Springer, Heidelberg (2007)
21. Li, N., Grosof, B., Feigenbaum, J.: Delegation Logic: A Logic-based Approach to Distributed Authorization. ACM Transactions on Information and System Security 6(1) (February 2003)
22. Li, N., Mitchell, J.: Datalog with constraints: A foundation for trust management languages. In: 5th International Symposium on Practical Aspects of Declarative Languages (2003)
23. Cooper, S., Taleb-Bendiab, A.: Concensus: multi-party negotiation support for conflict resolution in concurrent engineering design. Journal of Intelligent Manufacturing 9(2) (March 1998)

24. Querou, N., Rio, P., Tidball, M.: Multi-party negotiation when agents have subjective estimates of bargaining powers. Journal of Group Decision and Negotiation 16(5) (September 2007)
25. Czenko, M.R., Doumen, J.M., Etalle, S.: Trust management in P2P systems using standard TuLiP. In: 2008 Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Trondheim, Norway, May 2008, pp. 1–16 (2008)
26. Zhang, C.C., Winslett, M.: Multiparty trust negotiation: A new approach to distributed authorization. Technical Report UIUCDCS-R-2008-2976, Department of Computer Science, University of Illinois at Urbana-Champaign (2008)
27. Dijkstra, E.W., Scholten, C.S.: Termination detection for diffusing computations. Information Processing Letters 11(1), 1–4 (1980)
28. Winsborough, W.H., Li, N.: Towards practical automated trust negotiation. In: IEEE International Workshop on Policies for Distributed Systems and Networks (April 2002)
29. Winsborough, W.H., Li, N.: Safety in automated trust negotiation. ACM Transactions on Information and Systems Security 9(3), 352–390 (2006)
30. Frikken, K.B., Atallah, M.J., Li, J.: Attribute-based access control with hidden policies and hidden credentials. IEEE Transactions on Computers 55(10), 1259–1270 (2006)
31. Li, J., Li, N., Winsborough, W.H.: Automated trust negotiation using cryptographic credentials. In: ACM Conference on Computer and Communications Security, pp. 46–57 (2005)